DIFdoc: a standard format for visualizing hierarchical dataflow representations

Ivan Corretjer and Shuvra S. Bhattacharyya
Department of Electrical and Computer Engineering, and Institute for Advanced Computer Studies
University of Maryland, College Park, MD 20742, USA

Introduction

High performance embedded computing continues to push the boundaries of performance and complexity for modern embedded system design tools. Many tools for design of embedded digital signal processing (DSP) systems have been developed that increase programmer productivity by using dataflow models to represent the target application (e.g., see [2, 3, 7, 9]). For DSP-oriented applications, dataflow has been shown to be a highly intuitive conceptual and visual format.

As design complexity increases, so do the resulting dataflow graphs and one way to manage this complexity is through hierarchical decomposition of designs. Current DSP design tools usually present this hierarchy to designers via a recursive "look-inside" mechanism. Typically, the top-level graph containing hierarchical components is displayed in the current view and subsequent levels of the hierarchy are accessed by clicking through hierarchical components which then replace the current view. When working with complex designs involving several layers of hierarchy, this approach can lead to a loss of where in the hierarchy the present view is contained and how the current view fits into the overall application. Furthermore, the appearance of this approach is tool-dependent, and therefore it is not always possible to communicate such representations unambiguously outside a given design team.

In this abstract, we present a novel format for displaying hierarchical dataflow representations that addresses the problems described above. Our format presents the entire hierarchy to the designer in a tool-independent and intuitive format, while still allowing for convenient inspection of hierarchical components. Additionally, our format can be used to standardize dataflow graph descriptions for presentation and/or documentation purposes. To demonstrate our format we have integrated it into the dataflow interchange format (DIF) package.

DIF

The dataflow interchange format (DIF) project is an effort undertaken in the DSPCAD Research Group at the University of Maryland to standardize dataflow semantics, facilitate technology transfer for DSP design tools, and improve dataflow modeling and synthesis technology [5, 6]. DIF provides a programming language (called the 'DIF language' or just 'DIF') that unifies important forms of DSP-oriented dataflow modeling semantics, and also provides various components for the analysis and manipulation of graphs that are described in this language. Figure 1, which is adapted from [6], illustrates the role of DIF in DSP system design.

DIFdoc

We have developed a tool-independent, human-readable dataflow graph documentation format called DIFdoc for representing dataflow designs as hyperlinked combinations of HTML files and visual, graph representations. We have also developed a tool for generating DIFdoc representations from the DIF internal representation, which is the internal form to which all DIF language specifications are compiled into by the DIF language front-end.

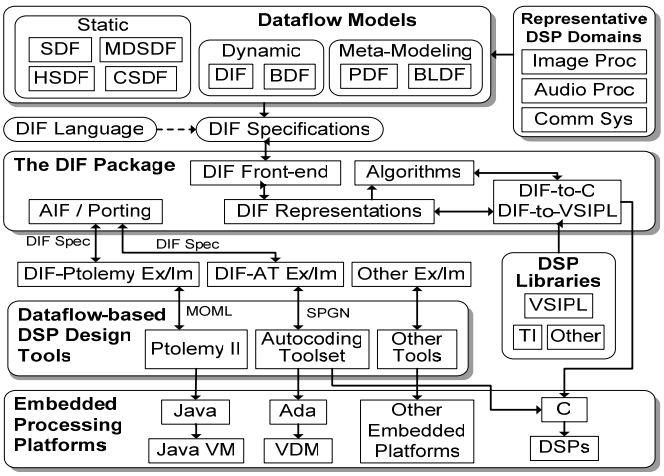

Figure 1: An illustration of the role of DIF in DSP design.

More specifically, the DIFdoc format displays dataflow-based design hierarchies using a combination of HTML (to represent hierarchical layers textually), and dot graphs (to represent individual graphs pictorially). The dot package is a well-known, freely-available software package for drawing graphs in their standard pictorial format [4]. This package employs sophisticated algorithms to draw graphs in ways that are visually intuitive.

The HTML portions of DIFdoc use indentation to represent the various layers of the design hierarchy, and hyperlinks so that for any level of the hierarchy, one has easy access to a visual (dot-based) representation for the dataflow graph at that level. Furthermore, deeply-nested designs can be organized through multiple HTML files, where the deepest levels of the hierarchy in the top-level representation are linked to separate HTML-based representations of their internal structures.

This hyperlinked, combined textual-pictorial representation of dataflow designs provides a unique representation of an entire dataflow hierarchy, a kind of view that is not available with conventional GUI-based dataflow tools, and is an especially convenient way for users to browse dataflow designs, and document them in a tool-independent way.

Application Examples

The DIFdoc representation of an FM demodulation application graph taken from the GNU radio software package [1] is depicted in Figure 2. The textual representation of the graph hierarchy can be seen in (a). The top-level graphical representation of the application is shown in (b), with the hierarchical component *demod* depicted in (c). The source signal is provided by the high speed analog-to-digital converter (ADC), and filter coefficients are generated to select the appropriate FM station to demodulate. By changing these coefficients through a user parameter, the same application graph can be used to demodulate any FM station. Once the FM station frequency has been selected, the signal (at FM frequencies) is downconverted to baseband, and passed to the demodulation subsystem. Inside *demod*, a quadrature demodulator operates on the baseband FM signal to extract a portion of the FM audio signal (namely, the left-plus-right audio information). This audio signal is then filtered before being sent to the PC sound card (audio sink).

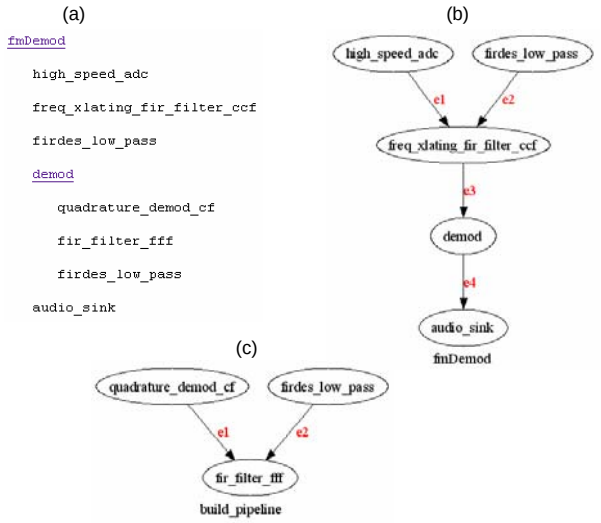

Figure 2: FM demodulation example from GNU radio.

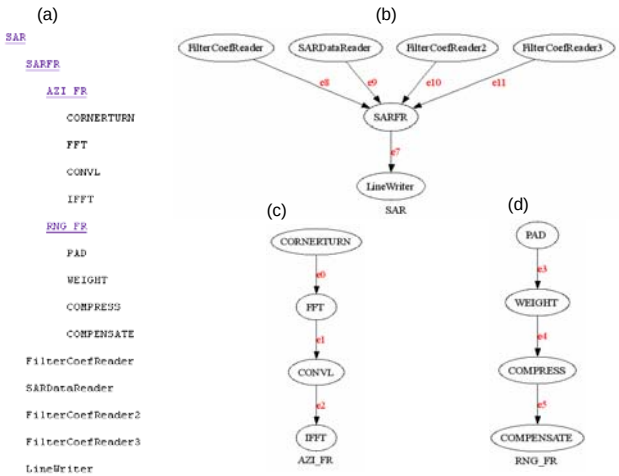

Figure 3: Synthetic aperture radar example from MCCI.

Another example is the synthetic aperture radar (SAR) application provided by MCCI [8], and shown in Figure 3 in DIFdoc format. Here, the application graph (b) and main processing components, Azimuth (c) and Range finding (d), are shown in graphical form after being accessed from the top-level textual representation (a).

Integration of DIFdoc and DIF

To facilitate adoption of our new DIFdoc format, we have integrated it into the DIF package. This was accomplished through the introduction of a new DIF attribute, *difdoc*. Users can specify several configuration parameters for the current DIF graph using the *difdoc* attribute, with a sample of these options summarized in Table 1.

Future work on DIFdoc involves the incorporation of interfaces to the hierarchy as well as providing for more detailed information to edge attributes. We plan on releasing an updated, full performance version of DIF with the DIFdoc enhancements in the near future.

Table 1: DIFdoc configuration parameters.

indentAmount	Sets the amount of indention for each level of hierarchy
maxIndent	Sets the maximum level of indention per HTML page
dotOutputFormat	Sets the format for dot generated graph files

Acknowledgements

This work is supported by the U.S. Defense Advanced Research Projects Agency through Management, Communications, and Control, Inc.

References

[1] E. Blossom. GNU radio: tools for exploring the radio frequency spectrum. *Linux Journal*, June 2004.

[2] J. Buck and R. Vaidyanathan. Heterogeneous modeling and simulation of embedded systems in El Greco. In *Proceedings of the International Workshop on Hardware/Software Co-Design*, May 2000.

[3] J. Eker et al., "Taming heterogeneity − the Ptolemy approach", *Proceedings of the IEEE*, January 2003.

[4] E. R. Gansner, S. C. North, E. Koutsofios, and K. Vo. A technique for drawing directed graphs. Technical report, AT&T Bell Laboratories, 1993.

[5] C. Hsu, F. Keceli, M. Ko, S. Shahparnia, and S. S. Bhattacharyya, "DIF: An interchange format for dataflow-based design tools", *Proceedings of the International Workshop on Systems, Architectures, Modeling, and Simulation*, pages 423-432, Samos, Greece, July 2004.

[6] C. Hsu and S. S. Bhattacharyya. Integrating VSIPL support in the dataflow interchange format. In *Proceedings of the Annual Workshop on High Performance Embedded Computing*, pages 75-76, Lexington, Massachusetts, September 2005.

[7] Lauwereins, R., Engels, M., Ade, M., and Peperstraete, J. A, "Grape-II: A system-level prototyping environment for DSP applications", *IEEE Computer Magazine*, 28(2):35-43, February 1995.

[8] C. B. Robbins, "Autocoding Toolset software tools for automatic generation of parallel application software", Technical report, Management, Communications, and Control, Inc., 2002.

[9] W. Thies, M. Karczmarek, and S. Amarasinghe. StreamIt: A language for streaming applications. In *Proceedings of the International Conference on Compiler Construction*, 2002.